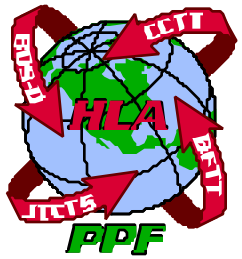


RTI Common Software Framework

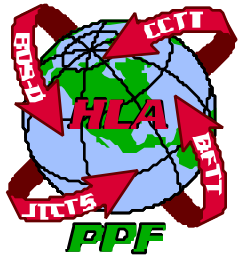
AMG-15

9 October 1996

TASC, Inc.

Michael Hooks, Rich Rybacki, Charles Koplik, Tony Lashley

Acknowledgments to: Stephen Bachinsky (SAIC) and
Chris Deschenes (I-Kinetics)



BACKGROUND

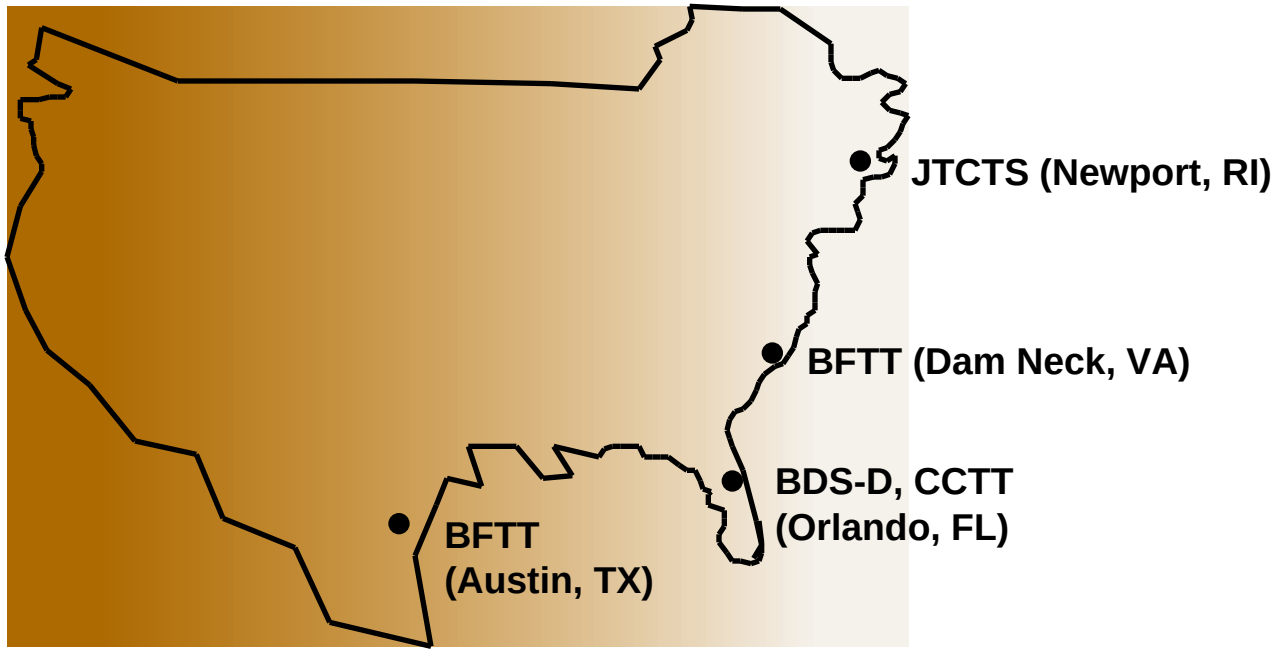
DMSO/STRICOM

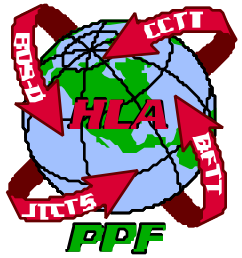
Platform Proto Federation Experiment

Platform Proto Federation

- BDS-D: M1 manned simulator
- BFTT: embedded simulators for carrier, destroyer, gunboat, and weapons
- CCTT: computer generated ground forces
- JTCTS: engineering models for live aircraft and weapons

HLA Evaluation Experiment Sites

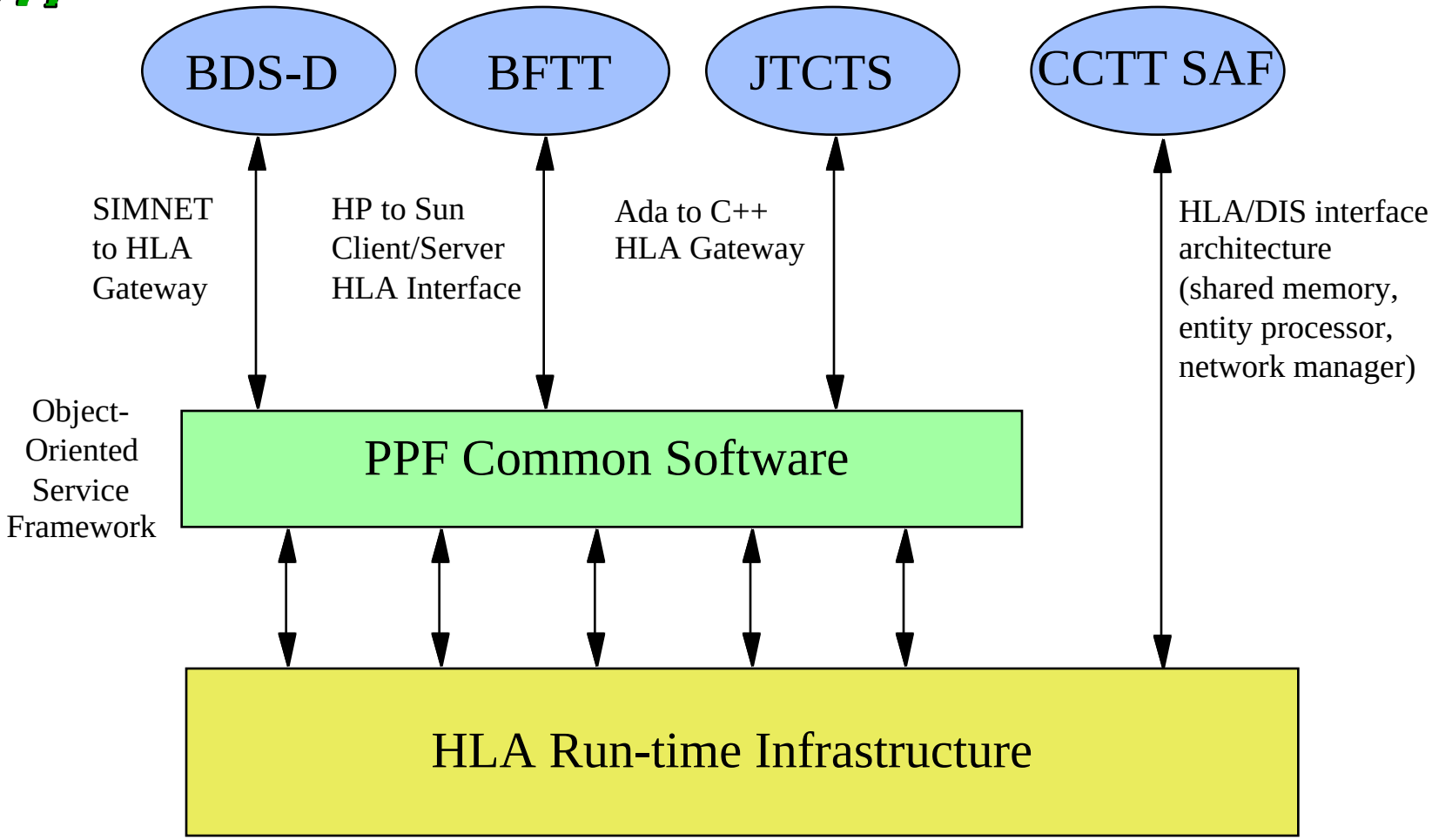


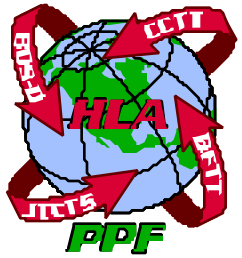


Objectives of Common Software Experiment

- Evaluate prototypical middleware for RTI
- Provide Federates with a set of service classes developed around a software framework
- Deliver to STRICOM the Federation Common Software
- Deliver User's Guide for the software

RTI Integration





Benefits of Middleware to HLA Community

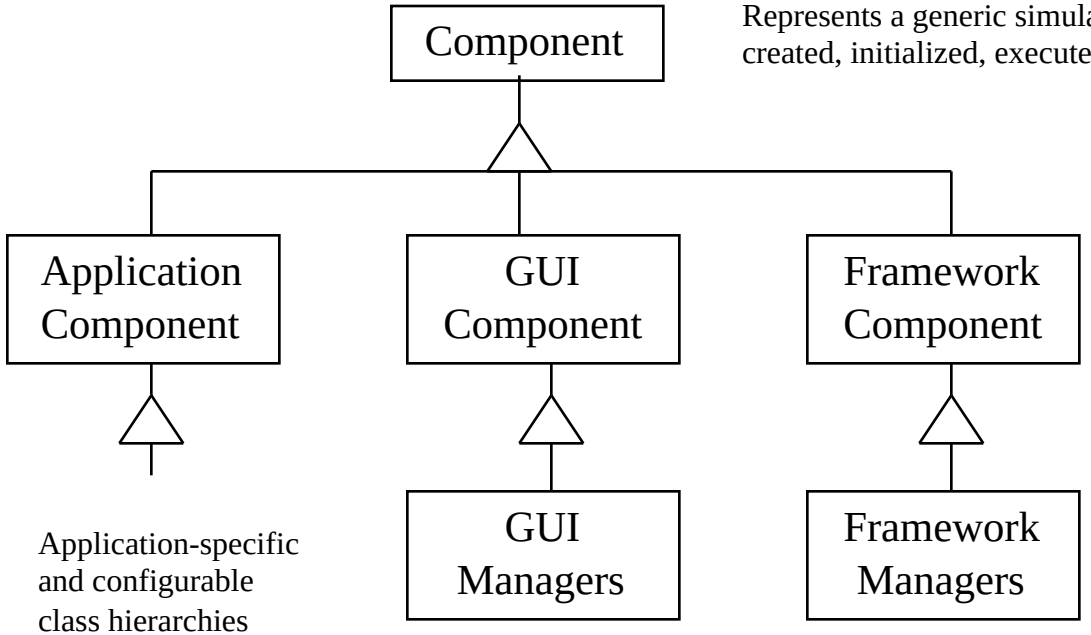
- Manage Complexity
  - RTI/HLA implementation issues solved in a central location
- Minimize Integration Time
  - Interface can be tailored to specific needs of the platform simulation
  - Results in lower development costs
- Maximize Extensibility
  - *Service Repository:* allows users to Plug and Play alternative implementations
  - *Object-Oriented Methodology* allows reusability of services by inheritance -- powerful new services can be created leveraging off of already existing services

APPROACH

Modular Software Framework

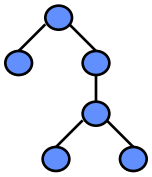
RTI Common Services

Modular Software Framework
Top Level Object Class Hierarchy



Represents a generic simulation object that can be created, initialized, executed, and managed.

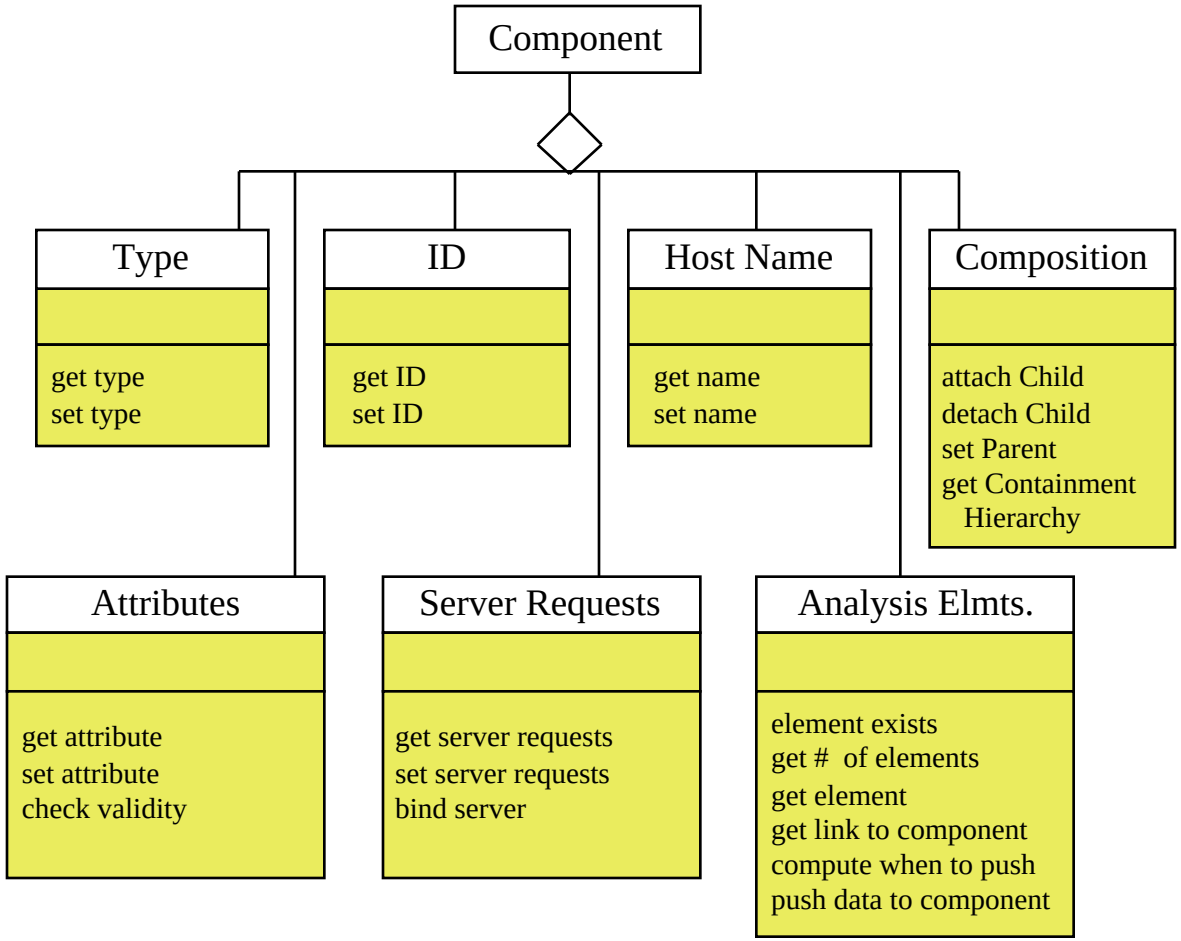
Application-specific and configurable class hierarchies



Collection of managers that provide services to support application-specific graphical user interfaces

Collection of component managers that provide services for a dynamic and reconfigurable simulation environment (access, configure, execute, control, modify, save)

Modular Software Framework Component Class



Type: classification mechanism used to define the interface capability of the component and the implementation name

ID: identification mechanism for uniquely distinguishing a particular component

Host: location where component exists (computer, cluster)

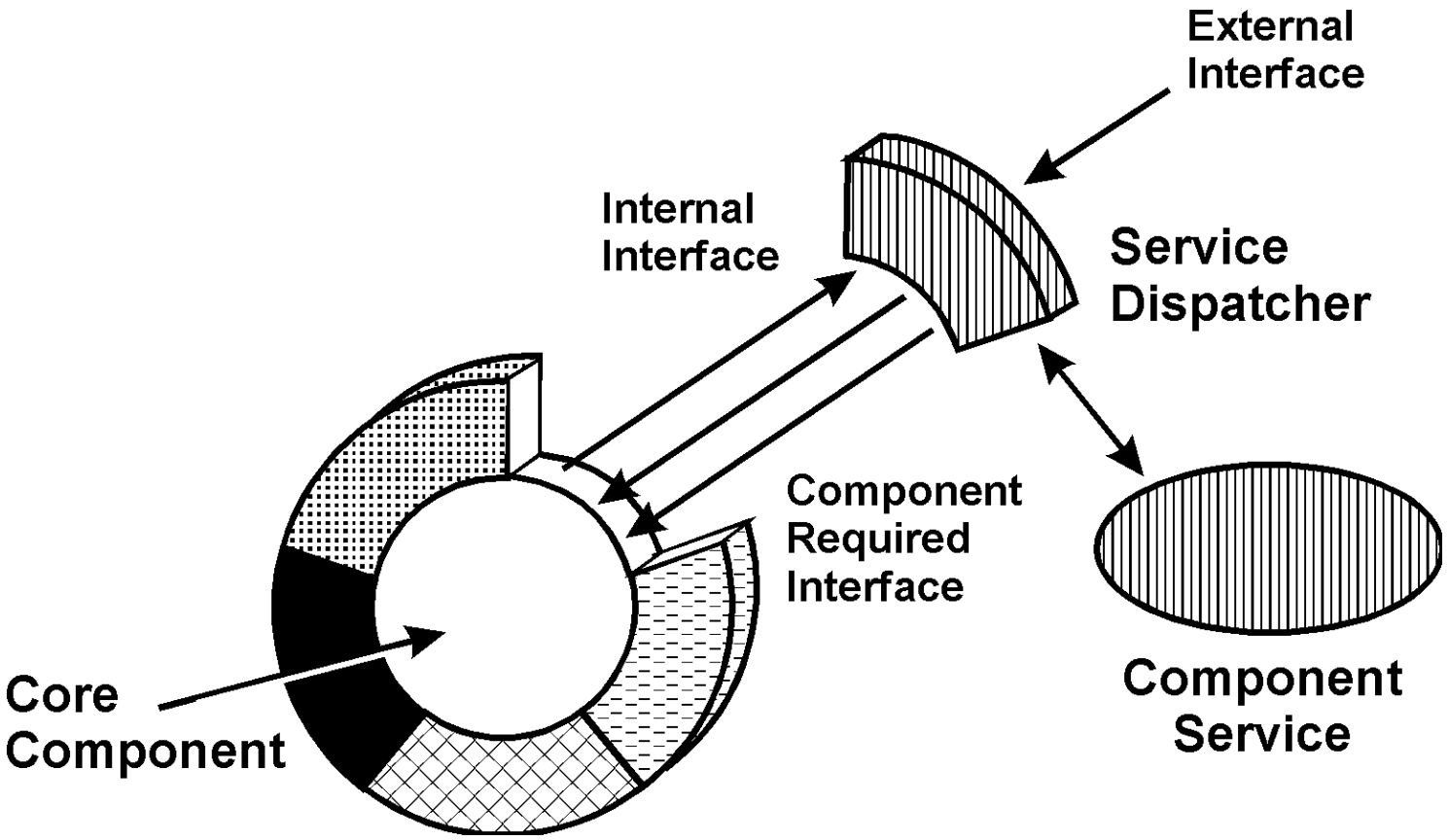
Attributes: definition and manipulation of model initialization parameters

Composition Manager: specifies notional hierarchy relationship

Analysis Elements: definition and selection of accessible data

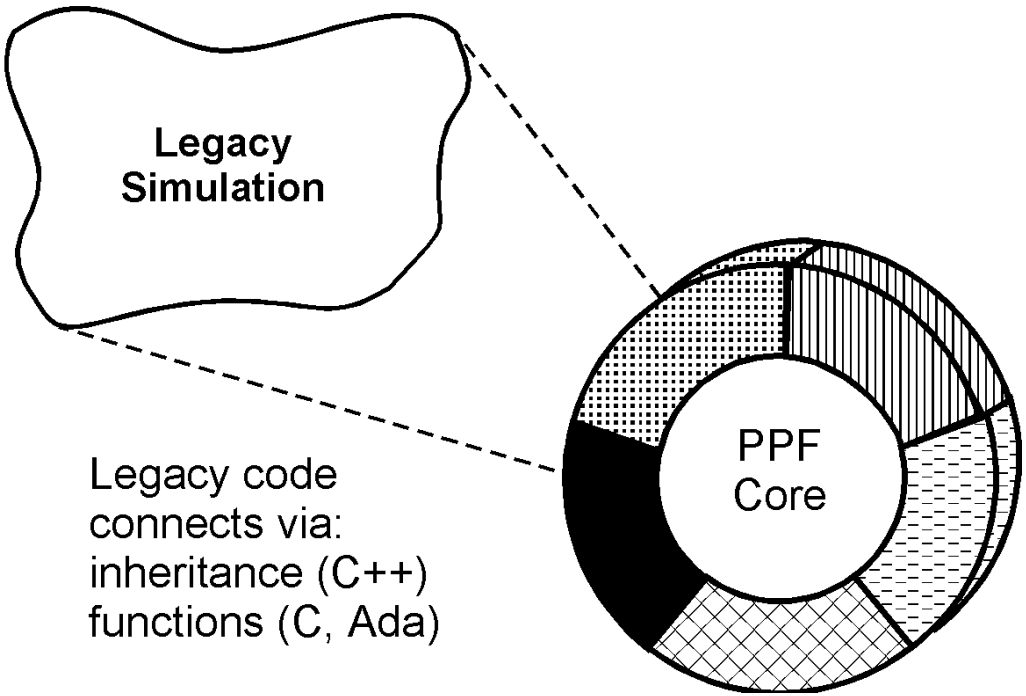
Server Requests: definition and access to servers required by component

Component Service Framework



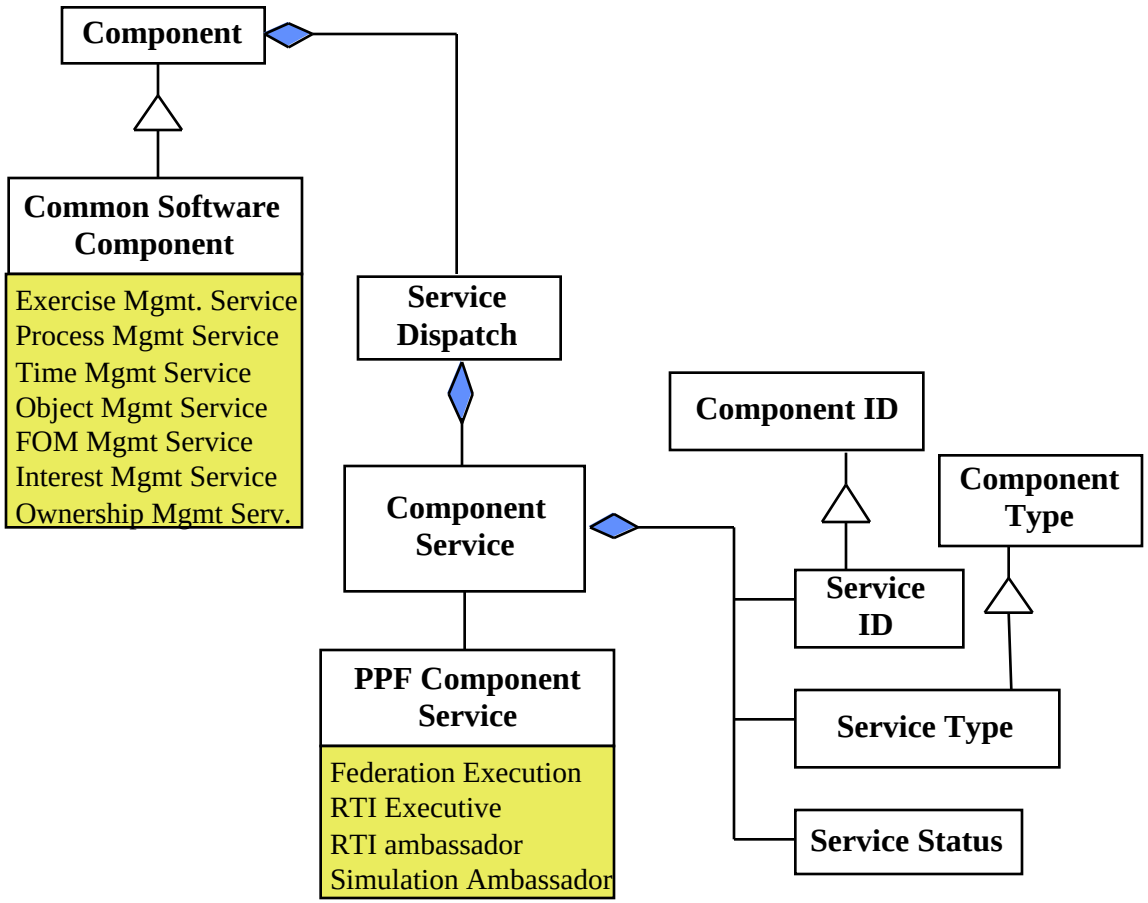
Interface to Legacy Systems

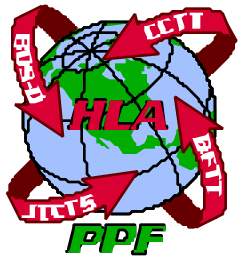
Simulation may represent multiple federation objects



Legacy code connects via:
inheritance (C++)
functions (C, Ada)

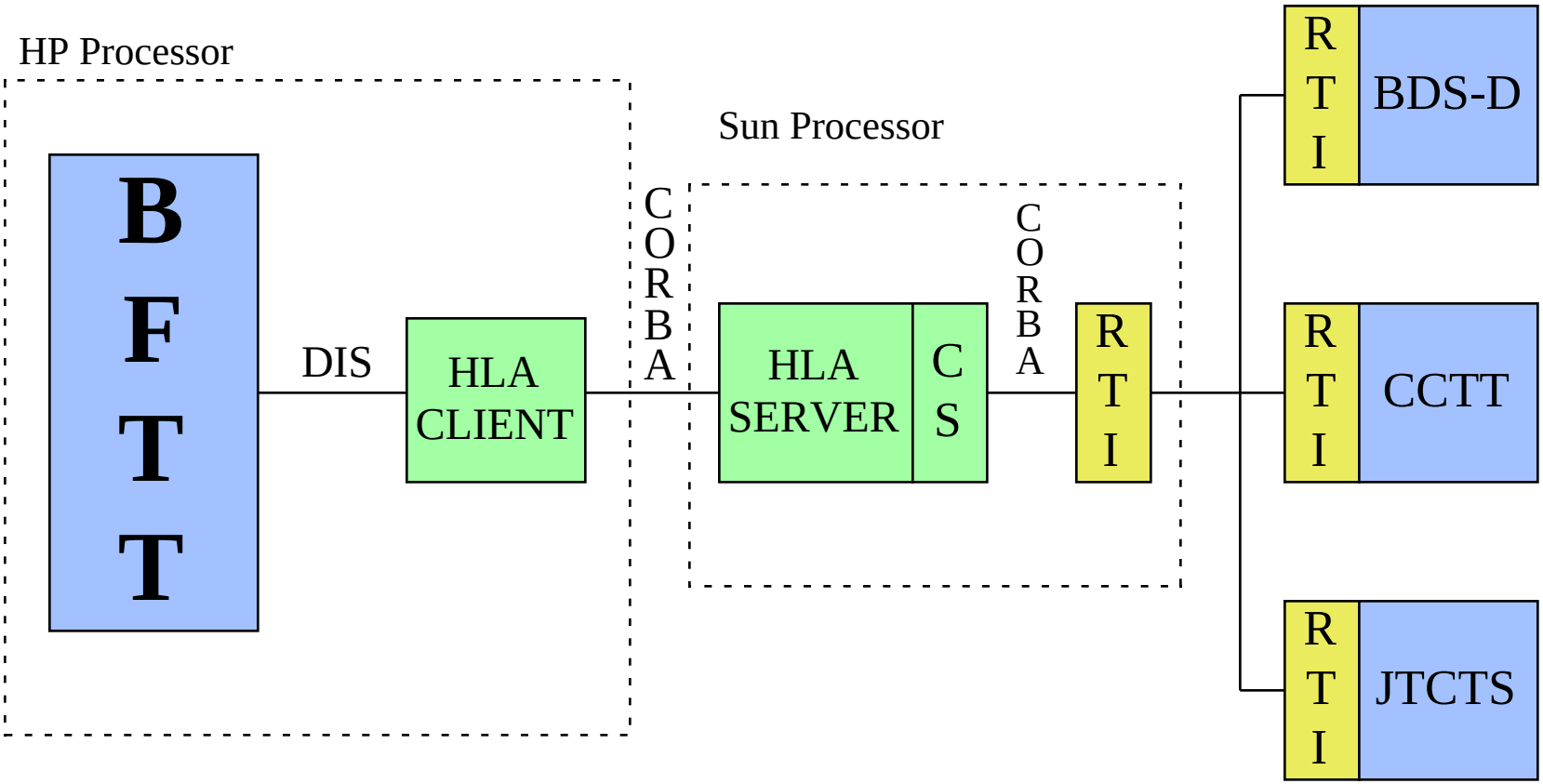
Component Service Class Diagram





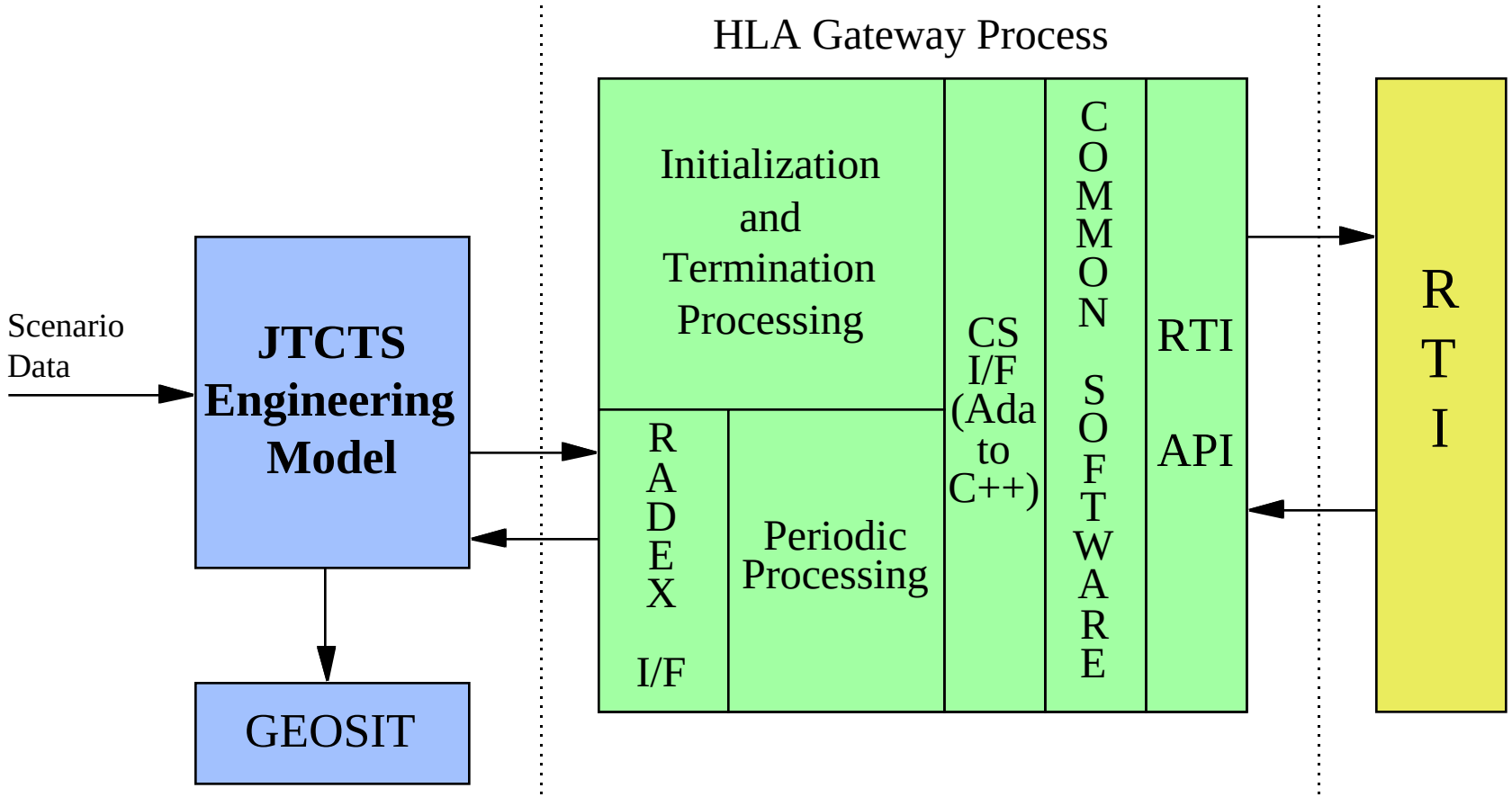
FEDERATE IMPLEMENTATION

BFTT Common Software Implementation

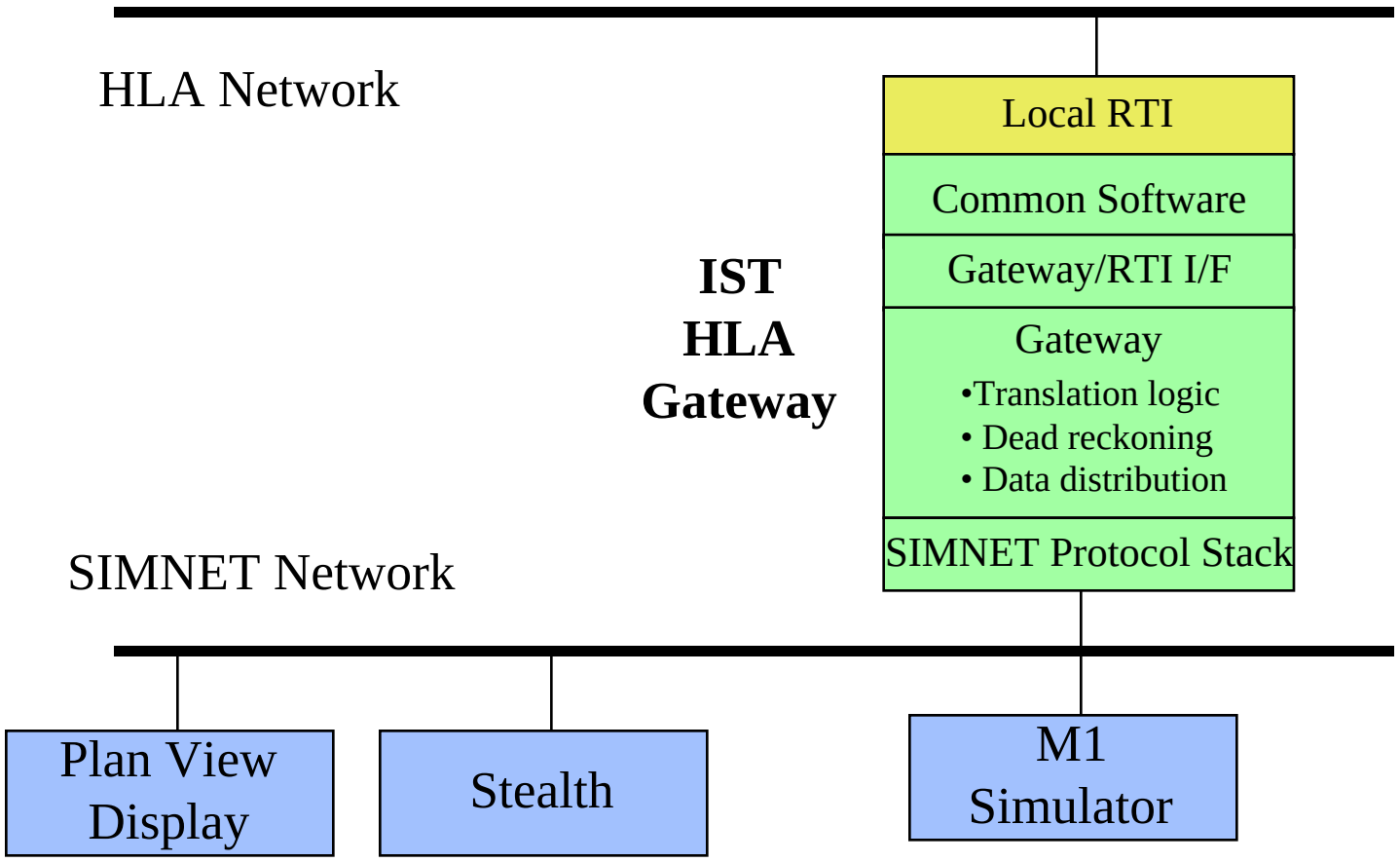


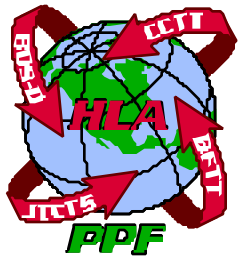
CS: Common Software

JTCTS Engineering Model Common Software Implementation

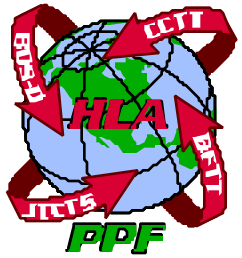


BDS-D Common Software Implementation



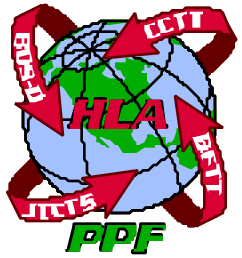


Services Provided



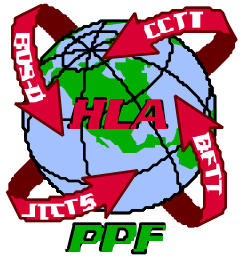
PPF Service Classes

- FOM Management
  - database of FOM class, object, attribute, & interactions relationships
- Federation Management
  - federation execution support (create, join, resign)
- Object Management
  - object, attribute, & interaction publication and transmission support
- Interest Management
  - object, attribute, & interaction subscription and reflection support
- Ownership Management
  - object and attribute ownership transferal support
  - federate synchronization support
- Time Management
  - federate synchronization support



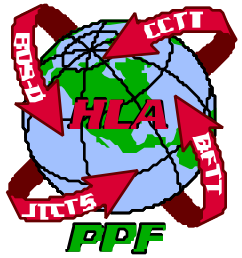
FOM Management Service Summary

- Maintains a database of active classes, attributes, and interactions within the particular federate
- Class definitions contain relationships to the attributes used to describe a FOM class. The FOM classes, attributes, and interactions use a dual identification mechanism consisting of a character string and a unique integer
- A database of active objects contains the RTI based ID for each object and provides the connection to the FOM class definition associated with the object



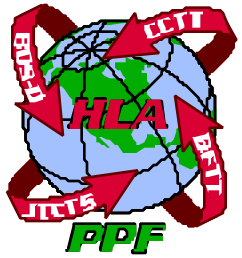
Federation Management Service

- Key Class Methods
  - createFederation
  - joinFederation
  - Obtain or set configuration information regarding host names, RTI server name, RID filename
- Configuration Behavior
  - Federation Execution easily customized through method interface, default arguments, or environment variables.
- RTI Abstraction
  - Simple abstraction to RTI federation management services.
  - Hides CORBA server binding operations.
  - Supports proper handling of RTI exceptions.



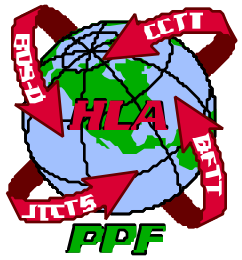
Object Management Service Summary

- Objects referenced by FOM class name and attributes
- Interactions are asynchronous events that can be sent along with a set of parameters at any time during the simulation.
- A database of published objects is maintained by the service class in order to simplify client operations
  - The Object Management Service is able to maintain the latest values of the attributes, only sending values to the RTI that have changed
  - The database also maintains the RTI update requirements for each object

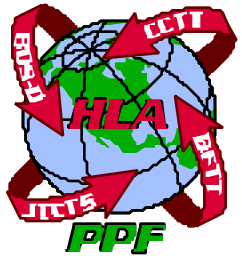


Interest Management Service Summary

- Subscription process instructs the RTI to only reflect attribute values or send interactions that match the interest declarations
- Reflected attribute values and interactions sent from other federates are buffered. The buffering mechanism maintains a collection of objects, and their corresponding attribute values, matching the interest criteria.
- Interactions sent to the federate are captured in a list. Clients of the Interest Management Service can access the object attribute and interaction parameter data using various query mechanisms.

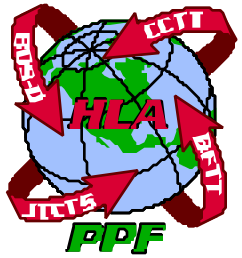


Lessons Learned



Federate Experience

- Common Software simplified development by providing needed common services (e.g., object database, asynchronous interaction) and insulating users from intricacies of RTI API and CORBA
  - However, parallel Federate, CS, and RTI development added delays and additional debugging complexity
  - Also, CS masking of RTI has drawbacks (one step away)
- BFTT was able to use Common Software as server running on Sun with client (and BFTT) on HP (saved porting). However, this does introduce latency
- BFTT implemented CS as a multi-threaded server with an HLA server for event processing and routing data. Solved problem of backlog in RTI event queue.



Federate Experience (continued)

- Polling as a means for Federate/CS interactions will not scale to large numbers of entities. Federate needs method for direct notification (BFTT)
- More efficient to provide common service to allow request of a block of entity ID's for future assignment (BFTT)
- Common Software attempt at generalization created inefficiencies (e.g., using strings to identify attributes instead of RTI handles to simplify implementation) (IST)
- Instrumentation of the PPF was easily achieved through changes to the Common Software (IST)